Recommender Systems
User-Facing Decision Support Systems

Michael Hahsler

Intelligent Data Analysis Lab (IDA@SMU)
CSE, Lyle School of Engineering
Southern Methodist University

EMIS 5/7357: Decision Support Systems
February 22, 2012
# Table of Contents

1. Recommender Systems & DSS
2. Content-based Approach
3. Collaborative Filtering (CF)
   - Memory-based CF
   - Model-based CF
4. Strategies for the Cold Start Problem
5. Open-Source Implementations
6. Example: recommenderlab for R
7. Empirical Evidence
"Decision Support Systems are defined broadly [...] as interactive computer-based systems that help people use computer communications, data, documents, knowledge, and models to solve problems and make decisions."

Power (2002)

Main Components:
- Knowledge base
- Model
- User interface
Recommender systems apply statistical and knowledge discovery techniques to the problem of making product recommendations. 

Sarwar et al. (2000)

**Advantages of recommender systems** (Schafer et al., 2001):
- Improve conversion rate: Help customers find a product she/he wants to buy.
- Cross-selling: Suggest additional products.
- Improve customer loyalty: Create a value-added relationship.
- Improve usability of software.
Recommender Systems

Recommender systems apply statistical and knowledge discovery techniques to the problem of making product recommendations. **Sarwar et al.** (2000)

**Advantages of recommender systems** *(Schafer et al., 2001)*:
- Improve conversion rate: Help customers find a product she/he wants to buy.
- Cross-selling: Suggest additional products.
- Improve customer loyalty: Create a value-added relationship.
- Improve usability of software.

→ A recommender system is a decision support systems which help a seller to choose suitable items to offer given a limited information channel.
Development Process

Prototyping
- Determine Objectives
- Develop
- Refine
- Demonstrate
- Test
- Implement

Spiral
- Analysis
- Evaluation
- Planning
- Development

Waterfall
- Requirements
- Design
- Implementation
- Verification
- Maintenance
Requirements

What problems is a recommender system supposed to solve?
What problems is a recommender system supposed to solve?

Important aspects:
- Available data
- Incentive structure
- Information channel
- Speed
- Quality of recommendations
- Trust
Recommendation strategies:

- **Content-based filtering**: Consumer preferences for product attributes.
- **Collaborative filtering**: Mimics word-of-mouth based on analysis of rating/usage/sales data from many users.

(Ansari et al., 2000)
# Table of Contents

1. Recommender Systems & DSS

2. Content-based Approach

3. Collaborative Filtering (CF)
   - Memory-based CF
   - Model-based CF

4. Strategies for the Cold Start Problem

5. Open-Source Implementations

6. Example: recommenderlab for R

7. Empirical Evidence
Content-based Approach

1. Analyze the objects (documents, video, music, etc.) and extract attributes/features (e.g., words, phrases, actors, genre).
2. Recommend objects with similar attributes to an object the user likes.
“The Music Genome Project is an effort to capture the essence of music at the fundamental level using almost 400 attributes to describe songs and a complex mathematical algorithm to organize them.”

http://en.wikipedia.org/wiki/Music_Genome_Project
Table of Contents

1. Recommender Systems & DSS
2. Content-based Approach
3. Collaborative Filtering (CF)
   - Memory-based CF
   - Model-based CF
4. Strategies for the Cold Start Problem
5. Open-Source Implementations
6. Example: recommenderlab for R
7. Empirical Evidence
Collaborative Filtering (CF)

Make automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many other users (collaboration).

Assumption: those who agreed in the past tend to agree again in the future.
Data Collection

Data sources:
- Explicit: ask the user for ratings, rankings, list of favorites, etc.
- Observed behavior: clicks, page impressions, purchase, uses, downloads, posts, tweets, etc.

What is the incentive structure?
Output of a Recommender System

- Predicted rating of unrated movies (Breese et al., 1998)
- A top-$N$ list of unrated (unknown) movies ordered by predicted rating/score (Deshpande and Karypis, 2004)
Types of CF Algorithms

- **Memory-based:** Find similar users (user-based CF) or items (item-based CF) to predict missing ratings.

- **Model-based:** Build a model from the rating data (clustering, latent structure, etc.) and then use this model to predict missing ratings.
Table of Contents

1. Recommender Systems & DSS
2. Content-based Approach
3. Collaborative Filtering (CF)
   - Memory-based CF
   - Model-based CF
4. Strategies for the Cold Start Problem
5. Open-Source Implementations
6. Example: recommenderlab for R
7. Empirical Evidence
User-based CF

Produce recommendations based on the preferences of similar users (Goldberg et al., 1992; Resnick et al., 1994; Mild and Reutterer, 2001).

1. Find \( k \) nearest neighbors for the user in the user-item matrix.
2. Generate recommendation based on the items liked by the \( k \) nearest neighbors. E.g., average ratings or use a weighting scheme.
User-based CF II

- **Pearson correlation coefficient:**

\[
\text{sim}_{\text{Pearson}}(x, y) = \frac{\sum_{i \in I} x_i y_i - I \bar{x} \bar{y}}{(I-1)s_xs_y}
\]

- **Cosine similarity:**

\[
\text{sim}_{\text{Cosine}}(x, y) = \frac{x \cdot y}{\|x\|_2 \|y\|_2}
\]

- **Jaccard index (only binary data):**

\[
\text{sim}_{\text{Jaccard}}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}
\]

where \( x = b_{ux} \), and \( y = b_{uy} \), represent the user's profile vectors and \( X \) and \( Y \) are the sets of the items with a 1 in the respective profile.

**Problem**
Memory-based. Expensive online similarity computation.
### Item-based CF

Produce recommendations based on the relationship between items in the user-item matrix (Kitts et al., 2000; Sarwar et al., 2001)

<table>
<thead>
<tr>
<th>S</th>
<th>(i_1)</th>
<th>(i_2)</th>
<th>(i_3)</th>
<th>(i_4)</th>
<th>(i_5)</th>
<th>(i_6)</th>
<th>(i_7)</th>
<th>(i_8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i_1)</td>
<td>-</td>
<td>0.1</td>
<td>0</td>
<td>0.3</td>
<td>0.2</td>
<td>0.4</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>(i_2)</td>
<td>0.1</td>
<td>-</td>
<td>0.8</td>
<td>0.9</td>
<td>0</td>
<td>0.2</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>(i_3)</td>
<td>0</td>
<td>0.8</td>
<td>-</td>
<td>0</td>
<td>0.4</td>
<td>0.1</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>(i_4)</td>
<td>0.3</td>
<td>0.9</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>0.3</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>(i_5)</td>
<td>0.2</td>
<td>0</td>
<td>0.7</td>
<td>0</td>
<td>-</td>
<td>0.2</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>(i_6)</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
<td>0.3</td>
<td>0.1</td>
<td>-</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>(i_7)</td>
<td>0</td>
<td>0.1</td>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>(i_8)</td>
<td>0.1</td>
<td>0</td>
<td>0.9</td>
<td>0.1</td>
<td>0</td>
<td>0.1</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

\(k=3\)  
\(u_a=\{i_1, i_5, i_8\}\)  
\(r_{ua}=\{2, ?, ?, 4, ?, ?, 5\}\)

**Recommendation:** \(i_3\)

1. Calculate similarities between items and keep for each item only the values for the \(k\) most similar items.
2. Use the similarities to calculate a weighted sum of the user's ratings for related items.

\[
\hat{r}_{ui} = \frac{\sum_{j \in s_i} s_{ij} r_{uj}}{\sum_{j \in s_i} |s_{ij}|}
\]

Regression can also be used to create the prediction.
Item-based CF II

Similarity measures:

- Pearson correlation coefficient, cosine similarity, jaccard index
- Conditional probability-based similarity (Deshpande and Karypis, 2004):

\[
\text{sim}_{\text{Conditional}}(x, y) = \frac{\text{Freq}(xy)}{\text{Freq}(x)} = \hat{P}(y|x)
\]

where \(x\) and \(y\) are two items, \(\text{Freq}(\cdot)\) is the number of users with the given item in their profile.

Properties

- Model (reduced similarity matrix) is relatively small \((N \times k)\) and can be fully precomputed.
- Item-based CF was reported to only produce slightly inferior results compared to user-based CF (Deshpande and Karypis, 2004).
- Higher order models which take the joint distribution of sets of items into account are possible (Deshpande and Karypis, 2004).
- Successful application in large scale systems (e.g., Amazon.com)
Table of Contents

1. Recommender Systems & DSS
2. Content-based Approach
3. Collaborative Filtering (CF)
   - Memory-based CF
   - Model-based CF
4. Strategies for the Cold Start Problem
5. Open-Source Implementations
6. Example: recommenderlab for R
7. Empirical Evidence
There are many techniques:

- **Cluster users** and then recommend items the users in the cluster closest to the active user like.
- Mine **association rules** and then use the rules to recommend items (for binary/binarized data)
- Define a **null-model** (a stochastic process which models usage of independent items) and then find significant deviation from the null-model.
- Learn a **latent factor model** from the data and then use the discovered factors to find items with high expected ratings.
Latent semantic indexing (LSI) developed by the IR community (late 80s) addresses sparsity, scalability and can handle synonyms ⇒ Dimensionality reduction.

Matrix Factorization

Given a user-item (rating) matrix $M = (r_{ui})$, map users and items on a joint latent factor space of dimensionality $k$.

- Each item $i$ is modeled by a vector $q_i \in \mathbb{R}^k$.
- Each user $u$ is modeled by a vector $p_u \in \mathbb{R}^k$.

such that a value close to the actual rating $r_{ui}$ can be computed (e.g., by the dot product also known as the cosine similarity)

$$r_{ui} \approx \hat{r}_{ui} = q_i^T p_u$$

The hard part is to find a suitable latent factor space!
Singular Value Decomposition (Matrix Fact.)

Linear algebra: **Singular Value Decomposition (SVD)** to factorizes $M$

\[ M = U \Sigma V^T \]

$M$ is the $m \times n$ (users $\times$ items) rating matrix of rank $r$. Columns of $U$ and $V$ are the left and right singular vectors. Diagonal of $\Sigma$ contains the $r$ singular values.
Linear algebra: Singular Value Decomposition (SVD) to factorizes \( M \)

\[
M = U \Sigma V^T
\]

\( M \) is the \( m \times n \) (users \( \times \) items) rating matrix of rank \( r \).
Columns of \( U \) and \( V \) are the left and right singular vectors.
Diagonal of \( \Sigma \) contains the \( r \) singular values.

A low-rank approximation of \( M \) using only \( k \) factors is straightforward.

The approximation minimizes approx. error \( ||M - M_k||_F \) (Frobenius norm).
Challenges (Matrix Fact.)

SVD is $O(m^3)$ and missing values are a problem.

1. Use Incremental SVD to add new users/items without recomputing the whole SVD (Sarwar et al., 2002).

2. To avoid overfitting minimize the regularized square error on only known ratings:

$$\text{argmin}_{p^*, q^*} \sum_{(u,i) \in \kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2)$$

where $\kappa$ are the $(u,i)$ pairs for which $r$ is known.

Good solutions can be found by stochastic gradient descent or alternating least squares (Koren et al., 2009).
1. For new user (item) compute $q_i \ (p_u)$.
2. After all $q_i$ and $p_u$ are known, prediction is very fast:

$$\hat{r}_{ui} = q_i^T p_u$$
1. Recommender Systems & DSS
2. Content-based Approach
3. Collaborative Filtering (CF)
   - Memory-based CF
   - Model-based CF
4. Strategies for the Cold Start Problem
5. Open-Source Implementations
6. Example: recommenderlab for R
7. Empirical Evidence
Cold Start Problem

What do we recommend to new users for whom we have no ratings yet?

Recommend popular items
Have some start-up questions (e.g., "What are your 10 favourite movies?")

What do we do with new items?
Content-based filtering techniques.
Pay a focus group to rate new items.
Cold Start Problem

What do we recommend to new users for whom we have no ratings yet?

- Recommend popular items
- Have some start-up questions (e.g., "What are your 10 favourite movies?")
Cold Start Problem

What do we recommend to new users for whom we have no ratings yet?

- Recommend popular items
- Have some start-up questions (e.g., "What are your 10 favourite movies?")

What do we do with new items?
Cold Start Problem

What do we recommend to new users for whom we have no ratings yet?

- Recommend popular items
- Have some start-up questions (e.g., ”What are your 10 favourite movies?”)

What do we do with new items?

- Content-based filtering techniques.
- Pay a focus group to rate new items.
Table of Contents

1. Recommender Systems & DSS
2. Content-based Approach
3. Collaborative Filtering (CF)
   - Memory-based CF
   - Model-based CF
4. Strategies for the Cold Start Problem
5. Open-Source Implementations
6. Example: recommenderlab for R
7. Empirical Evidence
Open-Source Implementations

- **Apache Mahout**: ML library including collaborative filtering (Java)
- **C/Matlab Toolkit for Collaborative Filtering**: (C/Matlab)
- **Cofi**: Collaborative Filtering Library (Java)
- **Crab**: Components for recommender systems (Python)
- **easyrec**: Recommender for Web pages (Java)
- **LensKit**: CF algorithms from GroupLens Research (Java)
- **MyMediaLite**: Recommender system algorithms. (C#/Mono)
- **RACOFI**: A rule-applying collaborative filtering system
- **Rating-based item-to-item recommender system**: (PHP/SQL)
- **recommenderlab**: Infrastructure to test and develop recommender algorithms (R)

Table of Contents

1. Recommender Systems & DSS
2. Content-based Approach
3. Collaborative Filtering (CF)
   - Memory-based CF
   - Model-based CF
4. Strategies for the Cold Start Problem
5. Open-Source Implementations
6. Example: recommenderlab for R
7. Empirical Evidence
100k MovieLense ratings data set: The data was collected through movielens.umn.edu from 9/1997 to 4/1998. The data set contains about 100,000 ratings (1-5) from 943 users on 1664 movies.

```
R> library("recommenderlab")
R> data(MovieLense)
R> MovieLense
943 x 1664 rating matrix of class ‘realRatingMatrix’ with 99392 ratings.
R> train <- MovieLense[1:900] # used for training
R> u <- MovieLense[901]       # active user
R> u
1 x 1664 rating matrix of class ‘realRatingMatrix’ with 124 ratings.
R> as(u, "list")[[1]][1:5]
           5             3
           5             1
      Mr. Holland’s Opus (1995)
           5
```
R> # create a recommender with training data
R> r <- Recommender(train, method = "UBCF")
R> r
Recommender of type ‘UBCF’ for ‘realRatingMatrix’
learned using 900 users.
R> # create recommendation for active user
R> recom <- predict(r, u, n = 5)
R> recom
Recommendations as ‘topNList’ with n = 5 for 1 users.
R> as(recom, "list")[[1]]
[3] "It’s a Wonderful Life (1946)"
recommenderlab: Compare Algorithms

R> # prepare data for 4-fold cross-evaluation
R> scheme <- evaluationScheme(db, method = "cross", k = 4,
+    given = 10)
R> # specify algorithms and parameters
R> algorithms <- list(
+  'random items' = list(name = "RANDOM", param = NULL),
+  'popular items' = list(name = "POPULAR", param = NULL),
+  'user-based CF' = list(name = "UBCF",
+    param = list(method = "Cosine", nn = 50)),
+  'item-based CF' = list(name = "IBCF",
+    param = list(method = "Cosine", k = 50)))
R> # start evaluation (predict 1...50 items)
R> results <- evaluate(scheme, algorithms,
+    n = c(1, 3, 5, 10, 15, 20, 50))
R> # plot results
R> plot(results, annotate = c(1, 3), legend = "right")
# Table of Contents

1. Recommender Systems & DSS

2. Content-based Approach

3. Collaborative Filtering (CF)
   - Memory-based CF
   - Model-based CF

4. Strategies for the Cold Start Problem

5. Open-Source Implementations

6. Example: recommenderlab for R

7. Empirical Evidence
Empirical Evidence

Empirical results from www.LeShop.ch (Dias et al., 2008).

Red lines: updates of the model.

Indirect revenue: purchase of recommended products (categories) in subsequent visits.


References II


Thank you!

This presentation can be downloaded from: http://michael.hahsler.net/ (under “Publications and talks”)